Project By: Hritul Pardhi 18070123061 ENTC - 'A'

Description: In this program of sentimental analysis of hotel reviews, I have taken the data of reviews of hotel, in which reviews had a column of happy or not happy, so based on the words used in the review and whether the person was happy or not happy after his visit at that hotel, gives me the data as an input in this project to feed to sklearn for training.

After the modelling and creating this project: One can write a review and check it automatically whether the person is happy or not happy after his/her visit.

```
from google.colab import drive
drive.mount('/gdrive')
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdr
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
# Mounting Drive
import os
os.chdir('/gdrive/My Drive/')
```

Data Facts and Importing Libraries

```
import pandas as pd
```

```
df = pd.read_csv('train.csv')
```

```
df.shape
```

```
(38932, 5)
```

```
df.head()
```

User_ID	Description	Browser_Used	Device_Used	Is_Response

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38932 entries, 0 to 38931
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   User_ID       38932 non-null  object
 1   Description   38932 non-null  object
 2   Browser_Used  38932 non-null  object
 3   Device_Used   38932 non-null  object
 4   Is_Response   38932 non-null  object
dtypes: object(5)
memory usage: 1.5+ MB
```

```
df.describe().transpose()
```

	count	unique	top	freq
User_ID	38932	38932	id23295	1
Description	38932	38932	Stayed for recent convention. Perfect location...	1
Browser_Used	38932	11	Firefox	7367
Device_Used	38932	3	Desktop	15026
Is_Response	38932	2	happy	26521

Data Cleaning / Data Analysis

```
## Checking missing vales in data set and printing percentage of missing values
## for each column

count = df.isnull().sum().sort_values(ascending=False)
percentage = ((df.isnull().sum()/len(df)*100)).sort_values(ascending=False)
missing_data = pd.concat([count,percentage],axis=1, keys=['Count','Percentage'])

print('Count and percentage of missing values for the colums:')
missing_data
```

```
Count and percentage of missing values for the colums:
```

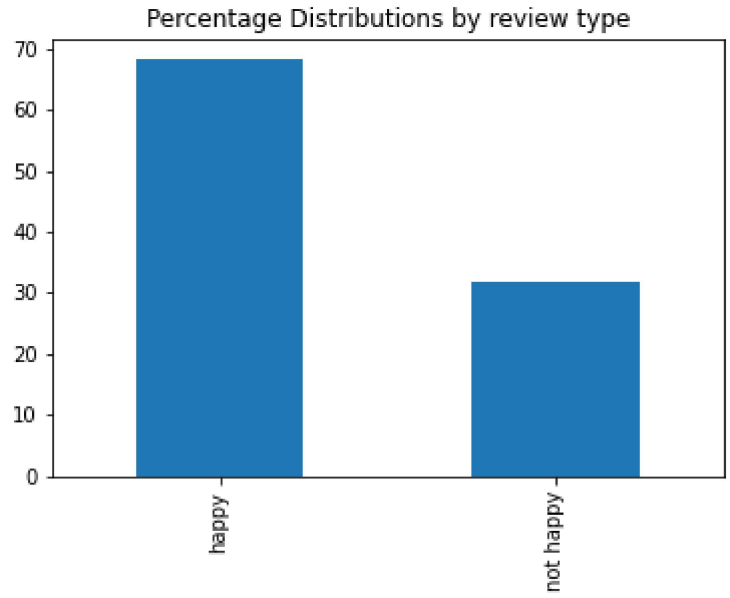
	Count	Percentage

```
## No missing values found
```

```
## checking for the distribution of default
import matplotlib.pyplot as plt
%matplotlib inline
print('Percentage for defalut\n')
print(round(df.Is_Response.value_counts(normalize=True)*100,2))
round(df.Is_Response.value_counts(normalize=True)*100,2).plot(kind='bar')
plt.title('Percentage Distributions by review type')
plt.show()
```

```
Percentage for defalut

happy        68.12
not happy    31.88
Name: Is_Response, dtype: float64
```




```
# Removing Extra columns
df.drop(columns=['User_ID','Browser_Used','Device_Used'],inplace=True)
```

```
df.head()
```

	Description	Is_Response
0	The room was kind of clean but had a VERY stro...	not happy

```
# Applying cleaning for text
import re
import string
# Function converts to lower case, removes square brackets, removes numbers and
# punctuations
def text_clean_1(text):
    text=text.lower() #lowercase
    text=re.sub('\[.*?\]','',text) #brackets
    text=re.sub('[%s]' % re.escape(string.punctuation),'',text) #punctuations
    text=re.sub('\w*\d\w*','',text) #digits
    text = re.sub('[''""...]','',text) #quotation mark
    text = re.sub('\n','', text) #next line/blank lines
    return text

cleaned1 = lambda x: text_clean_1(x)
```

```
df['cleaned_description'] = pd.DataFrame(df.Description.apply(cleaned1))
df.head()
```

	Description	Is_Response	cleaned_description
0	The room was kind of clean but had a VERY stro...	not happy	the room was kind of clean but had a very stro...
1	I stayed at the Crown Plaza April -- - April -...	not happy	i stayed at the crown plaza april april th...
2	I booked this hotel through Hotwire at the low...	not happy	i booked this hotel through hotwire at the low...
3	Stayed here with husband and sons on the	happy	stayed here with husband and sons on

MODEL TRAINING

```
from sklearn.model_selection import train_test_split

Independent_var = df.cleaned_description
Dependent_var = df.Is_Response

IV_train, IV_test, DV_train, DV_test = train_test_split(Independent_var, Dependent_var,

print('IV_train :', len(IV_train))
print('IV_test  :', len(IV_test))
print('DV_train :', len(DV_train))
print('DV_test  :', len(DV_test))
```

```
IV_train : 35038
IV_test  : 3894
```

```
DV_train : 35038
DV_test  : 3894
```

```
from sklearn.feature_extraction.text import TfidfVectorizer #importing essential librari
from sklearn.linear_model import LogisticRegression

tvec = TfidfVectorizer()
clf2 = LogisticRegression(solver = "lbfgs")


from sklearn.pipeline import Pipeline #for step by step execution
```

```
model = Pipeline([('vectorizer',tvec),('classifier',clf2)])

model.fit(IV_train, DV_train)


from sklearn.metrics import confusion_matrix

predictions = model.predict(IV_test)

confusion_matrix(predictions, DV_test)
```

```
array([[2417,  304],
       [ 154, 1019]])
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

print("Accuracy : ", accuracy_score(predictions, DV_test))
print("Precision : ", precision_score(predictions, DV_test, average = 'weighted'))
print("Recall : ", recall_score(predictions, DV_test, average = 'weighted'))
```

```
Accuracy :  0.8823831535695943
Precision :  0.8889271415963718
Recall :  0.8823831535695943
```

```
#Final output:
example = ["It was an wonderful experience, I would visit again"]
result = model.predict(example)

print(result)
```

```
['happy']
```